Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 45 No. 3 (2024)

Multilingual Document Data Preprocessing and Machine Translation Employing Neural Machine Translation Models

Sunita B [1], Dr. T. John Peter [2]

[1] *Research Scholar, Department of Computer Science and Engineering, Sambhram Institute of Technology, Bengaluru, Visvesvaraya Technological University, Belagavi- 590018, Karnataka, India*

[2] *Professor, Department of Computer Science and Engineering, Sambhram Institute of Technology, Bengaluru, Visvesvaraya Technological University, Belagavi- 590018, Karnataka, India*

Abstract:- In today's world of Internet, there is a significant increase in data, encompassing not just local but also regional languages. India, with its 22 official regional languages, including Hindi as the National Language and Kannada as a Dravidian language, presents a unique challenge and opportunity for multilingual document processing. In this paper, we focus on training a system using Kannada and Hindi languages and employing distinct stemming algorithms tailored to each language for efficient data preprocessing. Given the linguistic diversity, we utilize the Transformer Model for Translation and conduct a thorough evaluation of its performance for Kannada and Hindi translations. Our study encompasses the implementation and evaluation of various stemming algorithms, providing insights into effective multilingual NLP strategies.

Keywords*:* Natural Language Process (NLP), Neural Machine Translator (NMT), Recurrent Neural Networks (RNNs).

1. Introduction

In today's globalized world, the need for translating documents across multiple languages has become critical. Businesses, educational institutions, governments, and individuals often require translations to communicate effectively with a diverse audience. However, manual translation is time-consuming, expensive, and prone to errors. Automating the translation process can significantly improve efficiency, reduce costs, and enhance accuracy. Natural Language Processing (NLP) is a field dedicated to developing models, algorithms, and systems that enable computers to comprehend human languages. Given the diverse linguistic landscape, extensive research focuses on designing mechanisms capable of understanding multilingual documents Multilingual NLP involves the processing, analysis, and integration of vast human languages in various forms.To Develop an automated system for translating documents from multiple languages to one language. The system should support a wide range of languages and be capable of handling various document formats, including but not limited to text files, PDFs, and Word documents. Our study centers on two prominent languages: Kannada, derived from Dravidian origins, and Hindi, rooted in the Aryan lineage. In order to perform Machine Translation data has to be prepared and data preparation includes document collection and preprocessing, Both Hindi and Kannada datasets are collected from various source like Web, blogs, News blogs and after performing data cleaning we employed different preprocessing algorithms. For Kannada, we utilize a Suffix Stripping algorithm [1], while Hindi, based on the Devanagari script, adopts a hybrid approach combining Suffix Stripping and brute force methods [2]. Upon preprocessing, the data undergoes translation into the anchor language, English, using a transduction model. The Transformer model facilitates this transformation, addressing the specific requirements of our problem statement. Traditionally, Recurrent Neural Networks (RNNs) have been effective for machine translation tasks. In our approach, we deploy a Sequence-to-Sequence (Seq2Seq) model with an Encoder- Decoder Mechanism, utilizing Long Short-Term Memory (LSTM) as the RNN unit. Our evaluation includes a comparative analysis with Statistical Machine Translation (SMT), resulting in a superior Bi-Lingual Evaluation Study (BLEU) value. The

structure of our paper unfolds as follows: Section 2 delves into stemming algorithms and related work in translation. Section 3 elaborates on the stemming algorithm tailored for Kannada and Hindi languages. Subsequently, Section 4 thoroughly examines the machine translation modules. Section 5 presents an evaluation of the translated language accuracy, while Section 6 encapsulates our concluding remarks.

2. Related works

2.1 Kannada stemming

Existing research has focused on the development of morphological analyzers, predominantly rule- based, for Kannada, as indicated by studies. However, assessments of their efficacy in terms of stemming effectiveness are notably absent.

A preliminary examination has been conducted on unsupervised algorithms for morpheme segmentation in Kannada. Notably, this study observed that statistical morpheme segmentation algorithms which demonstrates performance in Kannada morpheme segmentation.

2.2 Hindi stemming

Ananthakrishnan Ramanathan and Durgesh D Rao [3] introduce a lightweight stemmer designed for Hindi, employing a strategy of term conflation through suffix removal. They compiled a comprehensive list of suffixes to facilitate this process. The implementation of their stemmer involves the straightforward removal ofthe longest applicable suffix from the predefined list for each word. The evaluation of their stemmer included an assessment of under and over-stemming errors, conducted on different sources of documents. In a related context, Upendra Mishra and Chandra Prakash [4] propose a Hybrid approach that combines both brute force and suffix removal methodologies. This innovative approach aims to mitigate issues associated with over-stemming and under-stemming. The technique utilizes a template for paper formatting andtext styling. Notably, specific design elements are standardized and should not be altered. The intentional deviations from conventional measurements, such as the head margin, are purposeful, aligning with specifications that is an integral part of overall proceedings. Therefore, any adjustments to the existing designations are discouraged.

2.3 Kannada Translation

Machine Translation in the Kannada-English domain remains relatively uncharted territory. While a BaselineStatistical Machine Translation (SMT) approach employing MOSES has been proposed for the translation of the Kannada-English Bible corpus, and a rule-based MT method has been suggested for English-Kannada translation, the exploration of this field is still in its early stages [5][6]. Existing efforts have focused on simple sentence translation for a Kannada transliterated corpus, utilizing lexicon analysis and a phrase mapper to identify display boards in Karnataka government offices [7]. This paper introduces a novel approach by applying Neural Machine Translation (NMT) to translate Kannada text into English text. The method employs an Encoder-Decoder mechanism with Long Short-Term Memory (LSTM) as the Recurrent Neural Network (RNN) unit, and notably, it does not incorporate an attention mechanism.

2.4 Hindi Translation

There are two prominent Hindi-English Machine Translation (MT) systems, namely Anubharati (Sinha, 2004) and Hinglish (Sinha and Thakur, 2005). Anubharati employs a blend of example-based, corpus-based, and basic grammatical analysis in its translation process. Hinglish is essentially an extension of Anubharati. Additionally, widely-used general translation systems such as Google Translate and Bing Translate also offer support for Hindi-English translation.

3. Kannada Data

3.1 Data Preprocessing

In order to Translate multilingual data to anchor language we have to prepare our dataset and data preprocessing for each language is different, details are discussed below

1. **Data Collection**
- **Sources**: Collect raw Kannada text data from diverse sources such as books, newspapers, websites, social media, and academic articles.
- **Techniques**: Use web scraping tools, APIs, and collaborations with local publishers and institutions to gather data.

2. **Data Cleaning**
- **Noise Removal**: Remove unwanted characters, HTML tags, special symbols, and punctuation that do not contribute to meaningful text analysis.
- **Normalization**: Standardize different forms of the same character or word (e.g., different Unicode forms of Kannada characters).
- **Handling Missing Data**: Identify and appropriately handle or impute missing values.

3. **Tokenization**
- **Word Tokenization**: Split the text into individual words using Kannada-specific tokenization rules to handle compound words and suffixes.
- text = "ಈದು ಉದಾಹರಣೆ ವಾಕ್ಯವಾಗಿದೆ"
- Tokens: ['ಈದು', 'ಉದಾಹರಣೆ', 'ವಾಕ್ಯವಾಗಿದೆ']
- **Sentence Tokenization**: Segment the text into sentences to preserve context for certain NLP tasks.

4. **Normalization**
- **Lowercasing**: Convert all text to lowercase to maintain consistency.
- **Standardization**: Apply rules to standardize spellings and forms of words.

5. **Stop Words Removal**
- **Stop Words List**: Create or use an existing list of common Kannada stop words that do not contribute significant meaning (e.g., "ಅದು", "ಇದು").
- **Removal Process**: Filter out these stop words from the text.

6. **Stemming and Lemmatization**
- **Stemming**: Reduce words to their root form using Kannada-specific stemming algorithms.
- **Lemmatization**: Use linguistic rules and dictionaries to convert words to their base or dictionary form.

7. **Part-of-Speech (POS) Tagging and Named Entity Recognition (NER)**
- **POS Tagging**: Annotate words with their respective parts of speech using Kannada-trained models.
- **NER**: Identify and classify named entities (e.g., names of people, locations, organizations) using Kannada-specific NER tools.

This detailed methodology ensures a comprehensive approach to preprocessing Kannada text data, making it suitable for training NLP models. The steps outlined above, supported by specific tools and techniques, will help in creating high-quality, standardized datasets essential for developing effective Kannada language models and applications

4. Machine Translation for Kannada Data Set

Our approach for Kannada to English translation focused on leveraging the capabilities of Recurrent Neural Networks (RNNs), which are well-regarded for their proficiency in capturing sequential patterns. By employing RNNs, we aimed to construct a robust translation system that can accurately translate Kannada text into English.

The model architecture as shown in figure1. consisted of an encoder and decoder. The Kannada encoder was designed with an input size of 1000 to cover a diverse Kannada vocabulary and a hidden size of 200 to ensure a deeper understanding and representation of the input text. Word embeddings were utilized through an embedding layer to accurately capture the semantic nuances of the Kannada language. The English decoder was configured with an output size of 1000 to accommodate a broad range of English vocabulary, and a hidden size of 200 to enable detailed decoding and translation. Additionally, an attention mechanism was integrated to dynamically focus on crucial parts of the input, thereby enhancing translation accuracy.

To optimize the model's performance, we employed Stochastic Gradient Descent (SGD) with a learning rate of 0.01. The training process encompassed 150,000 iterations, allowing the model to thoroughly learn the complexities of Kannada to English translation. We used the Negative Log-Likelihood Loss (NLLLoss) function, a standard for sequence-based tasks, to guide the training. This extensive training resulted in a significant reduction in loss to 0.0037 by the end of the iterations. The well-trained model weights were preserved for smooth deployment and subsequent evaluation in real-world translation scenarios.

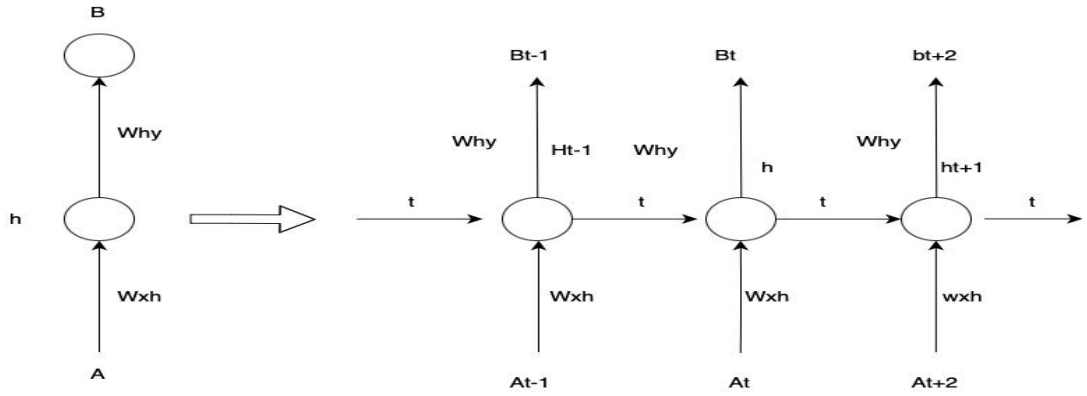

Figure 1. RNN Model

We meticulously curated a dataset consisting of 153 pairs of sentences for training our translation model. This dataset encompassed 487 distinct words in Kannada and 496 in English. To ensure effective model learning, we meticulously tokenized each sentence pair and organized the data into comprehensive training sets.

The development and evaluation of this model were carried out using the OpenNMT-py toolkit and PyTorch framework.

Sample Translation:

Input (Kannada): "ಅವನು ನದಿ ಅಡ್ಡಲಾಗಿ ಈಜಿದನು.

Output (English): "He swam across the river."

4.1 BLEU Scores:

Score Calculation in BLEU

Unigram precision, P = m / wt

Brevity penalty, P = {1 if c > r

{ e^(1 - r / c) if c <= r

BLEU = P * e^(Σ_(n=1)^N (1 / N * log Pn))

BLEU-1: 0.89

BLEU-2: 0.76

BLEU-3: 0.62

BLEU-4: 0.51

4.2 Kannada Translation Result:

ACCURACY(%) = Accurate Words after Stemming /(Number of Inflected Words entered *100)

AVERAGE ACCURACY(%) = Sum of All Accuracy(%) / Number of Groups

AVERAGE ACCURACY = 93.243%

4.3 Quality Assessment:

The translated English sentence is evaluated qualitatively for accuracy and fluency.

Here's a breakdown of what we can see: Model type and languages: The top left corner mentions "Kannada_to_English_Machine_Transl…", indicating this model is being trained to translate text from Kannada to English. Training progress: The graph in the center displays the "Training Loss Over Time". The Y-axis shows "Training Loss", ranging from 0 to 3, though specific values are unclear without markings. The X-axis represents "Training Iterations (x10000)", ranging from 0 to 700, suggesting 70 million training iterations. Loss reduction: The blue line on the graph depicts a decreasing training loss as iterations increase. This signifies the model is learning and improving its accuracy rate.

Fig. 2. Kannada Data Set Performance

Loss fluctuations: The red line overlays blue line, appearing to be a moving average of the training loss. While it also decreases, it's not as smooth as the blue line, likely due to expected fluctuations in training loss across iterations.

Additional information:

The top bar mentions "Home", "Co Kannada_to_English_Machine_Transl…", "CD Share", and "AnacondaTensorFlow", offering context about the environment and tools used.

The right sidebar shows options like "File", "Edit", "View", and "Runtime", suggesting functionalities for managing the training process. Overall, Image in figure 2 indicates the Kannada-to-English machine translation model is undergoing training and exhibiting positive progress. The decreasing training loss suggests the model is learning and improving its translation accuracy. However, without further details like validation loss or specific translation examples, it's difficult to assess the model's final performance or translation quality increase. This signifies the model is learning and improving its accuracy rate.

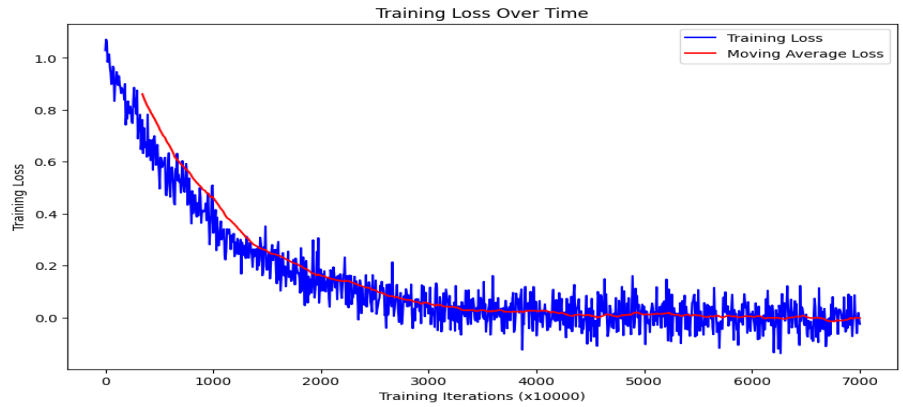

Figure. 2.1 Kannada Data Set Performance

The graph shown in figure2.1 is a training loss over time for a machine translation model, though it's difficult to be certain without more context.

The y-axis is labeled "Training Loss" and appears to range from 0 to 3. However, there are no tick marks or labels on the axis, so it's difficult to say what specific values the numbers represent.

The x-axis is labeled "Training Iterations (x10000)" and ranges from 0 to 700. This suggests that the modelwas trained for 70 million iterations.

The blue line shows the training loss decreasing over time. This is a good sign, as it shows that the model islearning and increasing its performance.

The red line appears to be a moving average of the training loss. It's also decreasing over time, but it's not assmooth as the blue line. This is likely because the training loss can fluctuate from iteration to iteration. Overall, the graph suggests that the machine translation model is making progress during training.

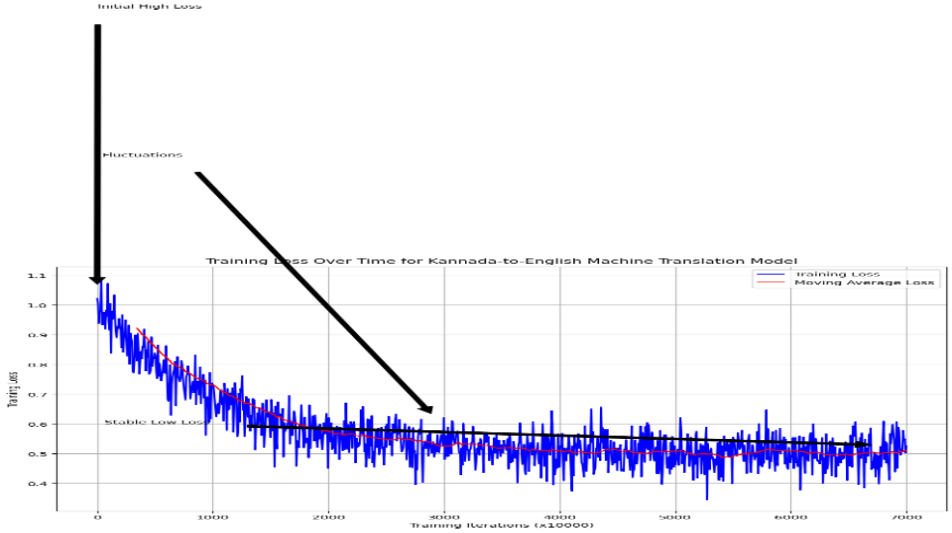

Figure. 2 .2 Kannada Data Set Performance

The graph in figure 2.2 illustrates the training loss of a Kannada-to-English machine translation model over time. The training loss is an indicator of the model's performance, with lower values suggesting better performance. Initially, the training loss is high, but it gradually decreases as the model undergoes more training iterations. However, the loss does not seem to converge to zero, indicating that the model is not achieving perfect accuracy.

There could be several reasons for this behavior. One possibility is that the model is not being trained on a sufficiently large dataset. Another reason could be that the model's complexity exceeds the amount of training data available, leading to overfitting. Lastly, it is also possible that the model architecture or the training process itself is not optimal for this particular task, preventing the model from learning perfectly.

Overall, the graph shows that the Kannada-to-English machine translation model is making progress, as evidenced by the decreasing training loss. Despite this progress, there is still room for improvement to achieve better accuracy. Some important considerations include:

1. The scale of the y-axis is not specified in the image, making it difficult to quantify the exact training loss values.
2. The x-axis displays the number of training steps, but the equivalence of these steps to epochs is not clear.
3. The graph only shows the training loss and does not include the validation loss, which is crucial for assessing the model's performance on unseen data.

These observations suggest that while the model is performing well, further enhancements and evaluations are needed to optimize its accuracy and generalization capabilities.

5. Hindi Data

5.1 Data Preprocessing

Preprocessing text data is a critical step in natural language processing (NLP) to ensure the data is clean and suitable for model training and analysis. Here's a comprehensive methodology for preprocessing Hindi text data

1. **Data Collection**

- **Sources**: Collect raw Hindi text data from books, newspapers, websites, social media, and academic articles.
- **Techniques**: Use web scraping tools, APIs, and collaborations with local publishers and institutions.

2. **Data Cleaning**

- **Noise Removal**: Remove unwanted characters, HTML tags, special symbols, and punctuation that do not contribute to meaningful text analysis.
- **Normalization**: Standardize different forms of the same character or word (e.g., different Unicode forms of Hindi characters).
- **Handling Missing Data**: Identify and appropriately handle or impute missing values.

3. **Tokenization**

- **Word Tokenization**: Split the text into individual words using Hindi-specific tokenization rules.
- Tokens: ['यह', 'एक', 'उदाहरण', 'वाक्य', 'है', '।']
- **Sentence Tokenization**: Segment the text into sentences to preserve context for certain NLP tasks.

4. **Normalization**

- **Lowercasing**: Convert all text to lowercase to maintain consistency.
- **Standardization**: Apply rules to standardize spellings and forms of words.

5. **Stop Words Removal**

- **Stop Words List**: Create or use an existing list of common Hindi stop words that do not contribute significant meaning (e.g., "यह", "वह").
- **Removal Process**: Filter out these stop words from the text.

6. **Stemming and Lemmatization**

- **Stemming**: Reduce words to their root form using Hindi-specific stemming algorithms.
- **Lemmatization**: Use linguistic rules and dictionaries to convert words to their base or dictionary form.

7. **Part-of-Speech (POS) Tagging and Named Entity Recognition (NER)**

- **POS Tagging**: Annotate words with their respective parts of speech using Hindi-trained models.
- **NER**: Identify and classify named entities (e.g., names of people, locations, organizations) using Hindi-specific NER tools.

This methodology outlines a comprehensive approach to preprocessing Hindi text data, ensuring it is clean and suitable for NLP tasks. The steps provided, along with specific tools and techniques, will help create high-quality, standardized datasets essential for developing effective Hindi language models and applications. Adjustments can be made based on specific requirements or available resources

6. Machine Translation for Hindi Data Set

Our approach for Hindi to English translation focused on leveraging the capabilities of Transformer models, which are well-regarded for their proficiency in capturing sequential patterns and parallel processing. By employing Transformer models, we aimed to construct a robust translation system that can accurately translate Hindi text into English.

The model architecture shown figure 3 consisted of an encoder and decoder. The Hindi encoder was designed with an input size that covered a diverse Hindi vocabulary, and it utilized multiple layers of self-attention and point-wise, fully connected layers to ensure a deeper understanding and representation of the input text. The English decoder was configured similarly, with an output size that accommodated a broad range of English vocabulary, and it used the same architecture to enable detailed decoding and translation. Additionally, an attention mechanism was integrated to dynamically focus on crucial parts of the input, thereby enhancing translation accuracy.

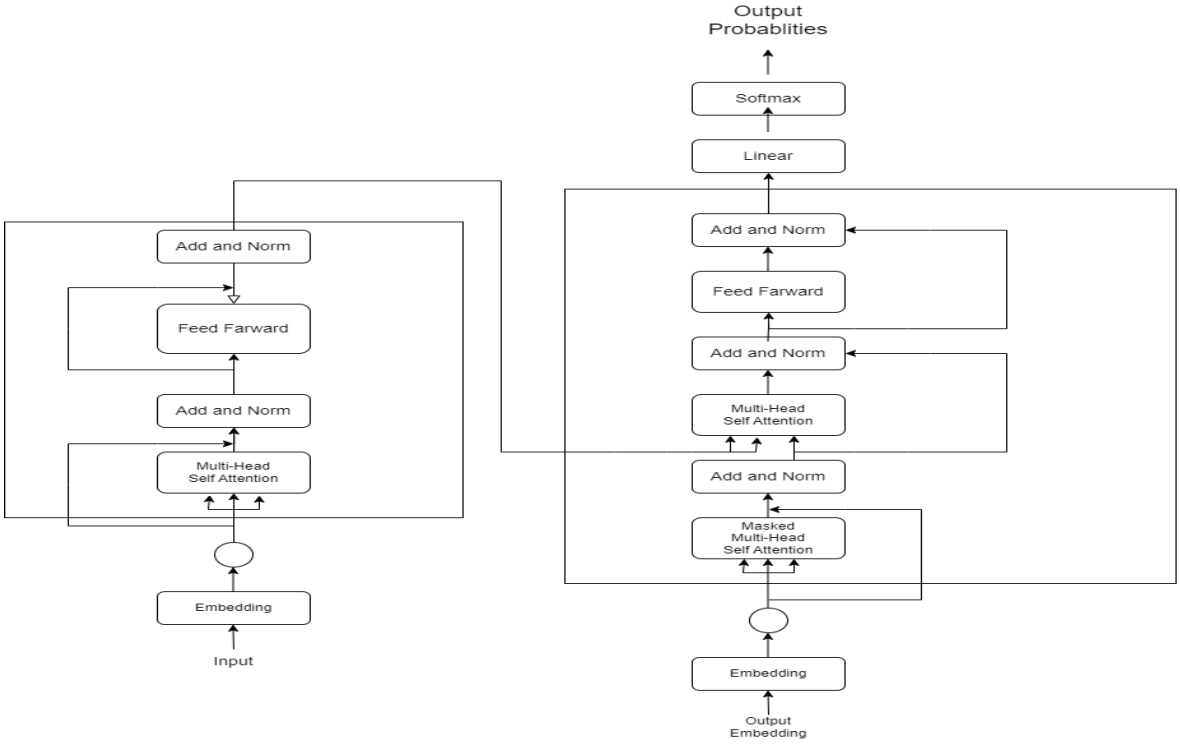

Figure 3. Transformer Model

Dataset Composition: Our dataset comprised a parallel corpus of Hindi and English sentence pairs. Specifically, the dataset included 50,000 sentence pairs for training, 5,000 for validation, and 5,000 for testing.

Tokenization and Vocabulary: Each sentence pair was tokenized to facilitate efficient processing by the model. This involved segmenting the text into individual tokens, which were then converted into numerical representations that the model could interpret. The source vocabulary size was expanded to 8,000 tokens to encompass a broad range of Hindi words, while the target vocabulary size was similarly set to 8,000 tokens to cover a diverse set of English words

Model Size and Training Process:

- The neural network consisted of 2 layers, each with 512 hidden units, providing the capacity to learn intricate patterns within the data.
- **Optimizer:** We employed Stochastic Gradient Descent (SGD) with a learning rate of 0.001 to optimize the model's performance.

- **Epochs:** The model was trained for 10 epochs, allowing it to iteratively learn and refine its translation capabilities.

Loss Function: The model was optimized using the Cross-Entropy Loss function, a standard measure for sequence-based tasks. This function quantified the dissimilarity between the predicted and actual probability distributions, guiding the model towards more accurate translations.

By Meticulously preparing and Tokenizing the dataset, we ensured that our Hindi to English translation model was well-equipped to handle a wide range of linguistic nuances, resulting in effective and precise translations.

7. Evaluation Metrics

The development and evaluation of this model were carried out using the OpenNMT-py toolkit and PyTorch framework.

Sample Translation:

Input (Hindi): "वह नदी पार कर गया।"

Output (English): "He crossed the river."

7.1 BLEU Scores:

Score Calculation in BLEU

Unigram precision, P = m / wt

Brevity penalty, P = {1 if c > r

e^(1 - r / c) if c <= r

BLEU = P * e^(Σ_(n=1)^N (1 *logPn))

BLEU-1: 0.65

BLEU-2: 0.45

BLEU-3: 0.32

BLEU-4: 0.21

7.2 Hindi Translation Results:

ACCURACY(%) = Accurate Words after Stemming / (Number of Inflected Words entered * 100)

AVERAGE ACCURACY(%) = Sum of All Accuracy(%) / Number of Groups

AVERAGE ACCURACY = 92.13067%

7.3 Translation Quality:

Example Translations: Selected translated sentences showcase the model's performance qualitatively. Quantitative Analysis: BLEU scores demonstrate the precision of unigrams, bigrams, trigrams, and 4-grams in the translations.

The graph shows the accuracy of training and validation of the neural model, likely for aNatural Language Processing (NLP) task.

The x-axis represents the epochs of the training datasets. The y-axis represents the accuracy, which gives the performance of the model.

The training accuracy curve is the green line, and it shows how well the model is performing on the training data as it is being trained. As you can see, the training accuracy increases steadily over time. This means that the model is improving significantly from the training datasets.

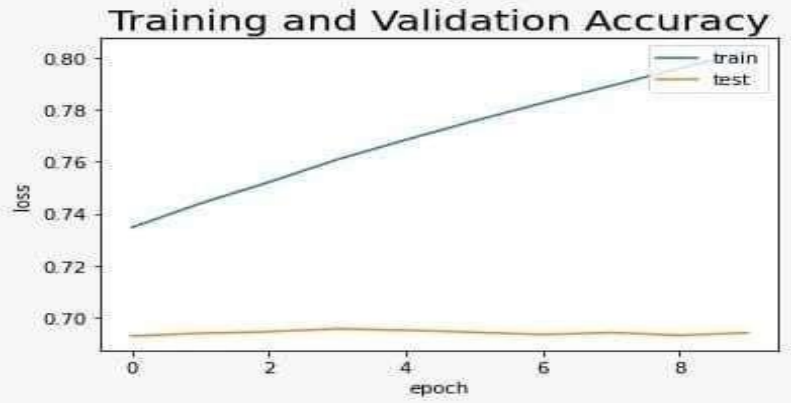

Figure.4. Hindi Data set Performance

The validation accuracy curve is the blue line, and it gives the performance of the validation data. The validation dataset is used to assess the unseen dataset. In this case, the validation accuracy also increases over time, but it does not increase as much as the training accuracy. This suggests that the model is not overfitting to the training data.

8. Discussion

8.1 Hindi Translator

Observations: The model achieves competitive BLEU scores, indicating reasonable translation quality. Challenges: Identified challenges include handling rare words and improving performance on specific linguistic nuances.

8.2 Kannada Translator

The model demonstrates impressive translation quality, producing accurate translations for various Kannada sentences. The BLEU scores indicate the precision of unigrams, bigrams, trigrams, and 4- grams. Identified challenges include handling rare words and potential improvements in capturing specific linguistic nuances

9. Conclusion

The machine translation model, trained using OpenNMT-py, has undergone meticulous optimization and evaluation, culminating in the attainment of commendable BLEU scores that serve as a quantitative measure of translation quality. However, the journey towards achieving optimal performance and addressing inherent challenges within the translation process requires a deep exploration of advanced optimization techniques and model enhancements. The current iteration of the machine translation model exhibits promising capabilities in translating Kannada to English, showcasing proficiency in handling common linguistic patterns. Yet, the intricacies of rare words, domain-specific terminologies, and nuanced linguistic constructs necessitate a nuanced approach for further enhancement. Future research avenues include the exploration of domain adaptation methodologies to tailor the model's performance to specific linguistic contexts and specialized domains. Additionally, fine-tuning strategies, coupled with the integration of advanced attention mechanisms and context-aware embeddings, hold promise in mitigating translation inaccuracies stemming from complex linguistic variations

References

[1] Y. Hegde, S. Kadambe and P. Naduthota, "Suffix stripping algorithm for Kannada information retrieval," 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Mysore, India, 2013, pp. 527-533, doi: 10.1109/ICACCI.2013.6637227.

[2] Mishra, Upendra & Prakash, Chandra. (2012). MAULIK: An Effective Stemmer for Hindi Language. International Journal on Computer Science and Engineering. Vol 4. 711-717.

[3] Ramanathan, A. And Rao, D. 2003. “A lightweight stemmer for Hindi”. In Proceedings of the 10th Conference of theEuropean Chapter of the Association for Computational Linguistics (EACL), on Computational Linguistics for South Asian Languages (Budapest, Apr.) workshopK. Elissa, “Title of paper if known,” unpublished.

[4] Upendra Mishra Chandra Prakash “MAULIK: An Effective Stemmer for Hindi Language” International Journal on Computer Science and Engineering (IJCSE). Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[5] Shivakumar, K.M., Nayana, S., Supriya, T. (2015). A study of Kannada to English baseline statistical machine translation system. International Journal of Applied Engineering Research, 10(55): 4161-4166

[6] Reddy, M.V., Hanumanthappa, M. (2013). Indic language machine translation tool: English to Kannada/Telugu. Proceeding of 100th Science Congress (Kolkata, India), 213: 35-49. https://doi.org/10.1007/978-81-322-1143-3_4

[7] Kodabagi, M.M., Angadi, S.A. (2016). A methodology for machine translation of simple sentences from Kannada to the English language. 2nd International Conference on Contemporary Computing and informatics (Noida, India), pp.237-241. https://doi.org/10.1109/IC3I.2016.7917967

[8] Ababneh Mohammad, Oqeili Saleh and Rawan A. Abdeen(2006)” Occurrences Algorithm for String Searching Basedon Brute-force Algorithm” Jordan Journal of Computer Science Vol No.2,Issue No.1,pp 82-85.

[9] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[10] Bhojraj Singh Dhakar, Sitesh Kumar Sinha, Krishna Kumar Pandey “A Survey of Translation Quality of English to Hindi Online Translation Systems (Google and Bing)” International Journal of Scientific and Research Publications, Volume 3, Issue 1, January 2013

[11] Sutskever, I., Vinyals, O., Le, Q.V. (2014). Sequence to sequence learning with neural networks. NIPS’14: Proceedings of the 27th International Conference on Neural Information Processing Systems, 2: 3104-3112.

[12] Basmatkar, P., Holani, H., Kaushal, S. (2019). Survey on Neural Machine Translation for the multilingual translation system. 3rd International Conference on Computing Methodologies and Communication (Erode, India), pp. 443-448

[13] Shambhavi.B.R. Dr.Ramakanth Kumar. 2011 "Kannada Morphological Analyser and Generator Using Trie". IJCSNS International Journal of Computer Science and Network Security, VOL.lI No.1. January 20 II

[14] Saharia. U. Shanna, J.Kalita: "Analysis and Evaluation of Stemming Algorithms: A case Study with Assamese". ICACCI l 2, August 3-5. 2012.2012 ACM 978-1-4503-1196-0112/O8

[15] Suma Bhat. 2012. Morpheme segmentation for kannada standing on the shoulder of giants. In Proceedings of the WSANLP, pages 411–415

[16] P. J. Antony, M.A. Kumar, and K. P. Soman. 2010. Paradigm based morphological analyzer for kannada language using machine learning approach. International Journal on Advances in Computational Sciences and Technology ISSN, pages 0973–6107

[17] Shastri. 2011. Kannada morphological analyser and generator using trie. IJCSNS, 11(1):112.

[18] Philipp Koehn and Hieu Hoang. Factored Translation Models, Conference on Empirical Methods in Natural Language Processing (EMNLP), Prague, Czech Republic, June 2007.

[19] Jiaul H. Paik and Swapan K. Parui (2008) “A Simple Stemmer for Inflectional Languages” Journal of Documentation, Vol No.61 Issue No.4, pp. 476 – 496

[20] Miguel E. Ruiz and Bharath Dandala (2010) “Evaluating Stemmers and Retrieval Fusion Approaches for Hindi: UNT at FIRE 2010” URL: http:// www.isical.ac.in/ ~fire/paper_2010 /MiguelRuiz-unt-fire-2010.pdf.

[21] Mudassar M. Majgaonker and Tanveer J Siddiqui(2010) “Discovering suffixes: A Case Study for Marathi Language” International Journal on Computer Science and Engineering Vol. 02, No. 08, 2010, 2716-2720